Appunti di Modelli matematici per le scelte di investimento

Applicazioni in MATLAB

1. Testi consigliati

Per approfondimenti si consigliano i seguenti testi:

- Glasserman P. *Monte Carlo Methods in Financial Engineering*, Springer;
- Brandimarte P. *Numerical Methods in Finance and Economics.* A MATLAB-Based Introduction. Wiley&Sons.

2. Esercizi in aula (comandi base)

```
%% vectors and matrices

x = 0.1:0.1:1          % declare a vector
x(1:3)                 % indexing a vector
x = repmat(1,2,10)

A = [2  3
     4  1]
B = [1  2
     3  1]
A.*B     % element by element product
A*B      % matrix multiplication

%% mean

A = [A
     0  0]
mean(A,1)
mean(A,2)

%% plot

x=[0:0.01:5];
y=2*x;
plot(x,y);          % plot

y2=exp(x);
plot(x,y,x,y2,'LineWidth',2);     % multiple plot

%% hist

```

```
32 x=randn(100,1); % generate a standard normal r.v.
33 hist(x);
34
35 %% indicator function
36
37 x= 0.1:0.1:1
38 indicator = x>0.5
39
40 %% find
41
42 x=randn(15,1)
43 a=find(x>1)
44 if length(a)==0
45     disp('Nessun numero maggiore di 1')
46 end
47 y=x(a)
48
49 %% diagrammi di flusso
50
51 %% sum the first 100 natural numbers
52
53 x=0;
54 for n=1:100
55     x=x+n;
56 end
57 x
58 y=sum(1:100)
59
60 %% cycle with WHILE
61
62 x=0;
63 n=100;
64 while n>0
65     x=x+n;
66     n=n-1;
67 end
68 x
69
70 %% for vs while
71
72 C = 0;
73 x = binornd(1,0.9);
74 while x==1
75     C = C+10;
76     x = binornd(1,0.9); % la condizione cambia durante il ciclo!
77 end
78 C
79
80 %% find the maximum value
81
82 x=randn(100,1);
83 m=-10000000;
84 for n=1:length(x)
85     if m<x(n)
```

```
86          m=x(n);
87      else
88      end
89  end
90  m
91  maximum=max(x)
92  clear all
93  clc
94
95  %% continue, break
96
97  % somma gli elementi >= 1
98  x = randn(10,1);
99  a = 0;
100 for n=1:10
101     if x(n)<1
102         continue    % passa all'iterazione successiva
103     else
104         a=a+x(n);
105     end
106 end
107 a
108 sum(x(find(x>1)))      % comando veloce
109
110 % somma i primi elementi fino a che non ne trovi uno > 1
111 a = 0;
112 for n=1:10
113     if x(n)>1
114         break   % termina il ciclo for
115     else
116         a=a+x(n);
117     end
118 end
119 a
120
121 %% generate a normal r.v.
122
123 % remember that X=sZ+m~N(m,s), dove Z~N(0,1)
124 Z=randn(10,1);
125 X=0.5*Z+1;
126
127 %% esercizio: dato un vettore x=randn(100,1), trovare l'elemento piu
        vicino ad 1.
128
129 x=randn(100,1);
130 d = 100000;
131 m=0;
132 for n=1:length(x)
133     a = abs(1-x(n));
134     if d>a
135         d=a;
136         m=n;
137     end
138 end
```

```
d
m

%% call an external function

z = L2getsquare(2);
% con "help getsquare" leggete l'help della funzione

%% input by user

y = input('Inserire un numero reale: ');
y = y^2

%% derivative

x = (0:0.1:1);
y = x.^2;
derivative = 2.*x;
h = 0.1;
approximation = ((x+h).^2-x.^2)/h;
error = abs(derivative-approximation);
clear all

%% integral

y = @(z) z.^2;
I = integral(y,0,4);

%% discrete integral

dx=0.1;
x = 0:dx:4;                 % dominio
y = x.^2;                   % funzione
II = trapz(y)*dx            % integrale trapezoidale
II_short = trapz(x,y)       % scrittura equivalente

x = dx:dx:2;                % discretizzazione per la prima parte del
    dominio
DX = dx/2;                  % discretizzazione piu precisa
a = 2+DX:DX:4;
x = [x a];
y = sin(x.^3)./x;
plot(x,y)
int1 = trapz(y)*dx
```

Nel precedente codice la funzione esterna *L2getsquare* è data dal seguente script:

```
function [ y ] = getsquare( x )
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

y=x.^2;

end
```

3. Somme di Riemann

```
% Evaluate Riemann's lower and upper sums

dx=0.01;
x = 0:dx:1;
y = x.^2;

lower = 0;
for n=2:length(x)
    lower = lower + dx*min(x(n-1).^2,x(n).^2);
end

upper = 0;
for n=2:length(x)
    upper = upper + dx*max(x(n-1).^2,x(n).^2);
end

distance = upper-lower
```

Con l'ultimo comando si valuta la distanza tra la somma superiore e quella inferiore.

Esercizio 3.1. *Scrivere una funzione che accetti come input una coppia di vettori (x, y) di uguale dimensione, in cui x rappresenta il dominio e y la funzione calcolata in quei punti. L'output è il valore dell'integrale, calcolato come media tra somma inferiore e somma superiore.*
Modificare la funzione appena creata in modo che accetti come input un parametro e, che rappresenta l'errore massimo che l'utente è disposto ad accettare.

4. Modello strutturale di Merton (1974)

```
%% PARAMETERS

T=10;          % orizzonte temporale
r=0.05;        % risk-free interest rate
B=1;           % debito
mu=0.1;
sigma=0.2;
V0=1;          % valore iniziale

N=100;
M=10000;
dt=T/N;

%% MERTON'S MODEL GENERATOR

rng(1);
V=zeros(M,N);     % inizializza V (allocazione memoria)
V(:,1)=V0;        % setta il valore iniziale per tutte le traiettorie

% versione non vettorializzata
tic
for m=1:M
    for i=1:N-1
        W=randn;
```

```
25            V(m,i+1)=V(m,i)+V(m,i).*(mu*dt+sigma*sqrt(dt)*W);
26        end
27    end
28    tempo1=toc;
29
30    % versione vettorializzata
31    tic
32    for i=1:N-1
33        W=randn(M,1);
34        V(:,i+1)=V(:,i)+V(:,i).*(mu*dt+sigma*sqrt(dt)*W);
35    end
36    tempo2=toc;
37
38    time=linspace(0,T,N);
39    plot(time,V(1:5,:),'k',time,repmat(B,N),'r','LineWidth',2);
40
41    %% DEFAULT TIME
42
43    tau=repmat(Inf,M,1);      % inizializza con "no default" (tau=infinito)
44    for m=1:M
45        for n=1:N
46            if V(m,n)<B
47                tau(m)=n;     % salva l'indice corrispondente al fallimento
48                break
49            end
50        end
51    end
52    tauY=dt*tau;      % trasforma gli indici salvati in anni
53
54
55    %% DEFAULT PROBABILITY
56
57    K=(log(B/V0)-(mu-0.5*sigma^2)*T)/(sqrt(T)*sigma);
58    P=normcdf(K);                          % formula esplicita
59    P_estimate=sum(V(:,end)<B)/M;          % approssimazione numerica
60
61    %% EQUITY AND DEBT VALUATION
62
63    % valutazione in un istante intermedio, conoscendo il valore V
64    c=zeros(M,1);
65    p=zeros(M,1);
66    t=2;              % istante di valutazione (in anni)
67    tt = t/dt+1;      % indice corrispondente all'istante t
68    [c,p] = blsprice(V(:,tt), B, r, T-t, sigma);
69    equity=c;
70    debt=exp(-r*(T-t))*B-p;
```

5. Bond con possibilità di default

```
1    %% PARAMETERS
2
3    T=10;          % orizzonte temporale
4    r=0.05;        % risk-free interest rate
5    gamma=0.1;
```

```
6
7  N=100;
8  dt=T/N;
9
10 %% DEFAULTABLE ZERO COUPON BOND
11
12 f= @(t) r+gamma*t;  % crea oggetto funzione (per integral)
13 I=0;                % I=1 default, I=0 no default
14 p = (1-I)*exp(-integral(f,0,T)) % prezzo
15
16 %% RECOVERY OF TREASURE
17
18 delta=0.5;          % loss given default
19 p = (1-delta)*exp(-r*T) + (1-I)*exp(-integral(f,0,T))
```

6. Credit Default Swaps

```
1  %% PARAMETERS
2
3  T=10;        % orizzonte temporale
4  r=0.05;      % risk-free interest rate
5
6  N=100;
7  dt=T/N;
8
9  %% risk-neutral implicit hazard rate
10
11 x_0 = 1;     % iterato iniziale
12 d=0.5;       % pagamento in caso di default
13
14 % f = V_prem - V_def
15 f = @(y) x_0*sum(dt.*exp(-(r+y)*(dt:dt:T)))-(d*y/(r+y))*(1-exp(-(r+y)
       *T));
16 [gamma,fxc,exitflag,output]=fzero(f,1);
17 gamma
18
19 %% CDS fair running spread premium
20 % determinare K* tale che p = 0
21
22 t=0;
23 A = @(t) 2*t+r;
24 h = @(s) integral(A,t,s);
25 H = @(s,K) (d*A(s)-K)*exp(-h(s));
26 g = @(K) integral(@(s) H(s,K),t,T,'ArrayValued',true);
27 [K_star,fxc,exitflag,output]=fzero(g,1);
28 K_star
29
30 %% gamma constant (verifichiamo l'algoritmo)
31
32 t=0;
33 A = @(t) 1+0*t+r;
34 h = @(s) integral(A,t,s);
35 H = @(s,K) (d*A(s)-K)*exp(-h(s));
36 g = @(K) integral(@(s) H(s,K),t,T,'ArrayValued',true);
```

```
[K_star,fxc,exitflag,output]=fzero(g,1);
K_star
teorico=d*(1+r)
```

7. Metodi Monte Carlo

```
%% a first Monte Carlo example

dx=0.01;      % domain discretization parameter
x = 0:dx:1; % domain discretization
y = x.^2;     % integrand function
a=x(1);
b=x(end);
M=1000;       % Monte Carlo parameter
 ascisse = unifrnd(a,b,M,1);
 indici = round(ascisse./dx);      % indici
 indici(find(indici==0))=1;        % assicuriamo che non ci siano indici
     nulli
 int2 = sum(y(indici))/M
 int1 = trapz(0:dx:1,(0:dx:1).^2)
 error = abs(int2-int1)

% alternative

 dx=0.01;                    % domain discretization parameter
M=1000;                      % Monte Carlo parameter
 a=0;
 b=1;
 x = unifrnd(a,b,M,1);       % domain discretization
 y = x.^2;                   % integrand function
 int2 = sum(y)/M
 int1 = trapz(0:dx:1,(0:dx:1).^2)
 error = abs(int2-int1)

% example 2

M=1000;                      % Monte Carlo parameter
 a=0;
 b=2;
 x = unifrnd(a,b,M,1);       % domain discretization
 y = exp(x);                 % integrand function
 int2 = b*sum(y)/M;
 int1 = exp(b)-1;
 error = abs(int2-int1)

% example 3

 a=0;
 b=10000;
M=b*1000;                    % Monte Carlo parameter
 x = unifrnd(a,b,M,1);       % domain discretization
 y = 0.5*exp(-0.5.*x);       % integrand function
 I = b*sum(y)/M;
 int1 = exp(-0.5*a)-exp(-0.5*b);
```

```
error = abs(I-int1)
```

8. Valutazione neutrale al rischio

```
%% PARAMETERS

T=10;          % time to maturity
r=0.05;        % risk-free interest rate
sigma=0.2;     % volatility
S0=1;          % initial stock price

N=100;         % time discretization
dt=T/N;        % interval length
M=5000;         % number of simulations

%% DISCOUNTED RISKY ASSET UNDER Q

rng(1);
S=zeros(M,N);
S(:,1)=S0;
for i=1:N-1
    W=randn(M,1);
    S(:,i+1)=S(:,i)+S(:,i).*(r*dt+sigma*sqrt(dt)*W);
end

% plot some trajectories
time=linspace(0,T,N);
plot(time,S(1:5,:),'k','LineWidth',2);

%% RISK-NEUTRAL VALUATION

K = 0.8;      % strike price

% Black & Scholes formula
[c,p] = blsprice(S0, K, r, T, sigma);

% Numerical approximation
% notice: T is the time to maturity
C = exp(-r*T)*mean((S(:,end)-K).*(S(:,end)>K));

error = abs(C-c);
```

9. Strategie di copertura alla Black-Scholes (ed errori di modello)

```
% HEDGING STRATEGIES
%% PARAMETERS

T=10;          % time to maturity
r=0.05;        % risk-free interest rate
sigma=0.2;     % volatility
S0=1;          % initial stock price
K=0.8;         % strike-price

N=500;         % time discretization
```

```
dt=T/N;       % interval length
M=1000;       % number of simulations

%% DISCOUNTED RISKY ASSET UNDER Q

rng(1);
S=zeros(M,N);
S(:,1)=S0;
for i=1:N-1
    W=randn(M,1);
    S(:,i+1)=S(:,i)+S(:,i).*(r*dt+sigma*sqrt(dt)*W);
end

%% HEDGING STRATEGY

delta = zeros(M,N-1);
delta = blsdelta(S(:,1:N-1),K,r,T,sigma);
[c,p] = blsprice(S0, K, r, T, sigma);

X = zeros(M,N);
X(:,1)=c;
for n=2:N
    X(:,n)=X(:,n-1)+delta(:,n-1).*(S(:,n)-S(:,n-1)) + (X(:,n-1)-delta
        (:,n-1)).*r.*dt;
end

meanerror = mean(abs( X(:,end)-(S(:,end)-K).*(S(:,end)>K) ));

%% what if S is not B&S?

% suppose S follows a CEV model
rng(1);
S=zeros(M,N);
S(:,1)=S0;
for i=1:N-1
    W=randn(M,1);
    S(:,i+1)=S(:,i)+S(:,i).*(r*dt+sigma.*S(:,i).^(0.1)*sqrt(dt).*W);
    %trovanan = find(isnan(S(:,i+1)));
    %S(trovanan,i+1) = S(trovanan,i)+S(trovanan,i).*r.*dt;
    % alternativa
    % trovanan = find(isnan(S));
    % S(trovanan) = S(trovanan-N)+S(trovanan-N).*r.*dt;
end
find(S<0);                    % some realizations may be negative
S(find(S<0))=-S(find(S<0));   % replace them!
find(S<0);                    % check out again

%% HEDGING STRATEGY

delta = zeros(M,N-1);
delta = blsdelta(S(:,1:N-1),K,r,T,sigma);
[c,p] = blsprice(S0, K, r, T, sigma);

X = zeros(M,N);
```

```
X(:,1)=c;
for n=2:N
    X(:,n)=X(:,n-1) + delta(:,n-1).*(S(:,n)-S(:,n-1)) + (X(:,n-1)-
        delta(:,n-1)).*r.*dt;
end

meanerrorCEV = mean(abs((X(:,end)-K)-(S(:,end)-K).*(S(:,end)>K)));
```

10. Stair-step diagram

```
%% Stair step diagram

f = @(x) sqrt(x);

dx=0.01;
plot([0:dx:3],f([0:dx:3]),[0:dx:3],[0:dx:3])

N=100;        % max number of repetitions
e = 0.01;     % max error
x = 0.1;        % initial iteration
for n=1:N
    if abs(f(x)-x)<e
        break
    else
        x = f(x);
    end
end
if n==N
    disp('Soluzione non trovata')
else
    x
end

% esempio 2

f = @(x) -x.^2-x;

plot([-3:dx:3],f([-3:dx:3]),[-3:dx:3],[-3:dx:3])

N=100;        % max number of repetitions
e = 0.01;     % max error
x = 1;  % initial iteration
for n=1:N
    if abs(f(x)-x)<e
        break
    else
        x = f(x);
    end
end
if n==N
    disp('Soluzione non trovata')
else
    x
end
```

```
45
46 % esempio 3
47
48 f = @(x) exp(x)-2;
49
50 plot([-3:dx:3],f([-3:dx:3]),[-3:dx:3],[-3:dx:3])
51
52 N=100;        % max number of repetitions
53 e = 0.01;     % max error
54 x = 0;  % initial iteration
55 for n=1:N
56     if abs(f(x)-x)<e
57         break
58     else
59         x = f(x);
60     end
61 end
62 if n==N
63     disp('Soluzione non trovata')
64 else
65     x
66 end
```

11. Assegnazione esercizi

```
1 %% Assegnazione esercizi per tesina finale
2
3 N = input('Numero studenti: ');
4 nomi = zeros(N,1);
5 nomi = string(nomi);
6 for n=1:N
7     nomi(n)=input('Nome: ','s');
8 end
9 x=randperm(6);
10 temp = 'Lo studente ';
11 for n=1:N
12     %assegnazione(n)=nomi(x(n));
13     message = strcat(nomi(n), ' svolge Es.', num2str(x(n)));
14     disp(message)
15 end
```